Exec

Objective

This projectlet will explore important concepts - the clock and calendar. In addition, will incorporate the basics of interprocess communication and multithreaded applications.

The utility exec takes a command line as input and executes it. Options to execute the command are:

- At a specified time today (or tomorrow if the time is already past)
- At specified intervals

There are several design choices. This could be a client server application whereby the server executes the commands. In this case this will be like cron.

We can also spin off a process for each command. In this approach each request creates a daemon that performs the actions at the required times. The daemons need to provide the means to query the status and/or interrupt their operations.

In either case some form of inter process communication solution is needed to ascertain the current status of all the outstanding requests and to abort them if necessary. Here again there are numerous choices present themselves.

Given the following specifications, there is scope for exploring the choices and combinations thereof.

Specifications

Command	Switch	Description
	—clock	Every time the clock hits the specified value.
	—timer	Execute at this period. The first execution starts after the specified time.
	Argument	Argument is the name of an executable / script. Without the switches —clock or —timer the command is run just once.
		Execution logs shall be created in the directory ~/logs with the name of the script forming the part of the log file name
status		List all the commands in the batch queue.
stop		Stop a particular instance of the job waiting

Example usage

Help

```
bin/exec

	Execute specified command according to the requested schedule

Usage:
  exec [command]

Available Commands:
  help        Help about any command
  run         Run the command
  status      Report the current status of all the jobs
  stop        Stop the execution of a command
  version     Report the version of the application

Flags:
      --config string   config file (default is $HOME/.cli/cli.yaml)
  -h, --help            help for exec
      --verbose         be verbose
  -v, --version         version for exec

Use "exec [command] --help" for more information about a command.
```

Example script

The following script is used in all the examples below:

```
#!/bin/bash

ping -c 5 www.google.com
```

Basic Operation

```
bin/exec run ./pinger.sh
2020/04/08 16:11:37 Invoking runner with args [./pinger.sh]
```

This generates the log file:

```
cat ~/logs/pinger.sh.log
-------------------------------------------------------------------------------
Command : ./pinger.sh
Started: Wed Apr  8 16:11:37 2020
-------------------------------------------------------------------------------
PING www.google.com (172.217.11.4): 56 data bytes
64 bytes from 172.217.11.4: icmp_seq=0 ttl=57 time=40.554 ms
64 bytes from 172.217.11.4: icmp_seq=1 ttl=57 time=16.608 ms
64 bytes from 172.217.11.4: icmp_seq=2 ttl=57 time=31.289 ms
64 bytes from 172.217.11.4: icmp_seq=3 ttl=57 time=11.874 ms
64 bytes from 172.217.11.4: icmp_seq=4 ttl=57 time=31.820 ms

--- www.google.com ping statistics ---
5 packets transmitted, 5 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 11.874/26.429/40.554/10.588 ms

-------------------------------------------------------------------------------
Terminated Wed Apr  8 16:11:41 2020 Duration 4.067297299s
————————————————————————————————————————————————
```

Run once after specified duration (10sec)

```
bin/exec run --timer 10s ./pinger.sh
2020/04/08 16:14:12 Invoking runner with args [./pinger.sh]
2020/04/08 16:14:12 Will wait till Wed Apr  8 16:14:22 2020
2020/04/08 16:14:22 Execution 1 at : Wed Apr  8 16:14:22 2020
2020/04/08 16:14:26 Repeat not requested. Quitting
```

Run periodically at the specified cadence (15sec)

In this example the specified script to be run at a cadence of 15 seconds. The runner then continues to execute till aborted with a ^C. In the meantime once in a while a status update is solicited and the printout shows the number of executions so far and the schedule for. The next execution. These queries are highlighted

```
$ bin/exec run --timer 15s --repeat ./pinger.sh
2020/04/08 16:15:56 Invoking runner with args [./pinger.sh]
2020/04/08 16:15:56 Asking for status updates
2020/04/08 16:15:56 Will wait till Wed Apr  8 16:16:11 2020
```

```
2020/04/08 16:15:56 Execution Count 0; Next execution at Wed Apr  8 16:16:11 2020
2020/04/08 16:16:11 Execution 1 at : Wed Apr  8 16:16:11 2020
2020/04/08 16:16:26 Execution 2 at : Wed Apr  8 16:16:26 2020
2020/04/08 16:16:41 Execution 3 at : Wed Apr  8 16:16:41 2020
2020/04/08 16:16:56 Execution 4 at : Wed Apr  8 16:16:56 2020
2020/04/08 16:17:11 Asking for status updates
2020/04/08 16:17:11 Execution 5 at : Wed Apr  8 16:17:11 2020
2020/04/08 16:17:15 Execution Count 5; Next execution at Wed Apr  8 16:17:26 2020
2020/04/08 16:17:26 Execution 6 at : Wed Apr  8 16:17:26 2020
2020/04/08 16:17:41 Execution 7 at : Wed Apr  8 16:17:41 2020
^C
```

Inspection of the ~/logs directory shows the creation of 1 log for each execution:

```
pinger.sh_1.log  pinger.sh_3.log  pinger.sh_5.log  pinger.sh_7.log
pinger.sh_2.log  pinger.sh_4.log  pinger.sh_6.log
```

Run to a clock ie when the timer reaches a particular value

```
bin/exec run --clock 16:27:01 ./pinger.sh
2020/04/08 16:26:11 Invoking runner with args [./pinger.sh]
2020/04/08 16:26:11 Still time for the minute
2020/04/08 16:26:11 Waiting till Wed Apr  8 16:27:01 2020
2020/04/08 16:27:11 Execution 1 at : Wed Apr  8 16:27:11 2020
2020/04/08 16:27:15 Repeats not requested. Quitting
```

In this modality, the repetition specifies to wait for the clock to reach the same value again - in 24 hours. But then if the specified clock had already expired for “today” then the first execution will wait for the clock to reach the same value the next day.

```
bin/exec run --clock 16:32:01 --repeat ./pinger.sh
2020/04/08 16:31:46 Invoking runner with args [./pinger.sh]
2020/04/08 16:31:46 Still time for the minute
2020/04/08 16:31:46 Waiting till Wed Apr  8 16:32:01 2020
2020/04/08 16:32:46 Execution 1 at : Wed Apr  8 16:32:46 2020
2020/04/08 16:36:46 Asking for status updates
2020/04/08 16:36:46 Execution Count 1; Next execution at Thu Apr  9 16:32:46 2020
```

Implementation Examples

Go language example

https://gitlab.com/RajaSrinivasan/exec.git

Potential Improvements

Limits	Some way to specify how to terminate
More clock values in the same day	Eg. At 6:00 am, 2:00 pm and 10:00 pm
status	List all the commands being executed now.
stop	Stop a particular job